CECS 327 Project 3

Distribution and Scalability

Ruben Bautista (29517824), Brian Ho (028695873) & Michael Jarrah (030328803)

May 3rd, 2025

Explanation of system design:

Architecture Overview:

Our system implements a decentralized peer-to-peer network with distributed hash table (DHT) routing for key-value storage. It consists of Docker-containerized nodes communicating over a virtual network. It also implements a gossip protocol to distribute information across nodes over time.

Key Components:

1. Nodes: Flask-based servers capable of file storage, key-value operations, and peer discovery
2. Bootstrap Node: Facilitates initial network discovery
3. DHT Routing: Distributes key-value pairs across nodes based on consistent hashing
4. Gossip Protocol: Enables decentralized information propagation with TTL limits (Bonus option)

Phase 1: File Upload/Download:

Local file storage with secure handlingREST API for file operations. The basic file operations are implemented using Flask's file handling capabilities. Files are stored locally in each node's storage directory. File uploads are secured using werkzeug.utils.secure_filename to prevent path traversal attacks. Content types are verified to permit only safe file extensions

Phase 2: Key-Value Store:

The key-value store is implemented as an in-memory dictionary within each node. Simple REST API endpoints for storing and retrieving values are present and keys and values can be any form of serializable data. Operations are atomic within a single node

Phase 3: DHT-Based Routing

- SHA-1 consistent hashing maps keys to responsible nodes
- Requests automatically forwarded to appropriate nodes
- Fallback mechanisms for resilience

Bonus Option 3: Gossip Protocol

- Propagates both peer information and key-value pairs
- TTL mechanism prevents message flooding
- Creates eventual consistency across nodes

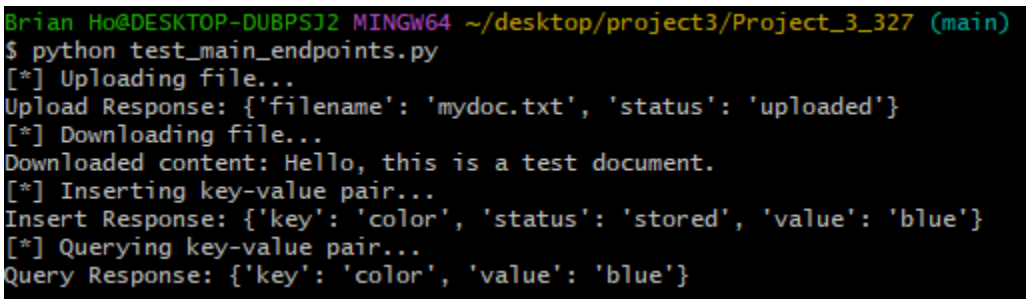
Technical Features

- Node Identification: Uses container hostnames and port mapping
- Request Forwarding: Routes requests to responsible nodes
- Error Handling: Implements fallbacks and retries
- Synchronization: Periodically exchanges data between peers

The system successfully demonstrates a functioning P2P network with DHT routing and resilient data distribution, fulfilling all project requirements while implementing the bonus gossip protocol extension.

Successful Runs:

File Upload/Download Run:



KV Storage Run:

```
(venv) michael@michaels-idea:~/code/uni/327/3$ python3 test_kv_store.py head -n 2 test_kv_store.py && tail -n 2 test_kv_store.py
node running at http://localhost:5001
testing key-value store functionality

test 1: storing key-value pair
stored: {'key': 'color', 'message': 'Value stored for key: color', 'status': 'success', 'storing_node': 5001, 'value': 'blue'}

test 2: retrieving stored value
retrieved: {'key': 'color', 'node': 5001, 'value': 'blue'}

test 3: retrieving non-existent key
got 404 as expected

test 4: storing multiple key-value pairs
stored: fruit = apple
stored: animal = tiger
stored: language = python

test 5: retrieving all values
got color = blue
got fruit = apple
got animal = tiger
got language = python

all tests passed
```

Video (https://youtu.be/VZ3mj8b9GN4)

Contributions:

Ruben Bautista: Coding, Testing, Writeup

Brian Ho: Coding, Testing, Writeup

Michael Jarrah: Coding, Testing, Writeup